AFRL
DoD Shared Resource Center
(DSRC)

SGI Altix 4700
User's Guide

January 20, 2010

Issued January 15, 2010

- Removed references to LSF
- Added PBSPro information
- Changed PAM information

Issued March 09, 2009

- Updated to reflect AFRL MSRC to AFRL DSRC

Issued January 20, 2008

- Reviewed and Updated to reflect the change from ASC MSRC to AFRL MSRC

Issued October 1, 2007

- Initial release.

Table of Contents

1.0 Introduction

This document provides an overview and introduction to using the SGI Altix 4700, located at the Air Force Research Laboratory (AFRL) Major Shared Resource Center (MSRC). The AFRL DSRC is located at Wright-Patterson Air Force Base, near Dayton, Ohio.

This guide is intended to provide information so customers who are familiar with the Linux operating system can create and run their own programs, as well as use existing software on the SGI Altix 4700.

1.1 Assumed Background of the Reader

It is assumed that the reader of this guide has a firm grasp of the concepts required to use the Linux operating system and to program in either the C, C++, FORTRAN 77, FORTRAN 90 or FORTRAN 95 languages.

1.2 New Terminology Introduction

The SGI Altix 4700 is a Dual-Core machine; there may be terminology not previously used with AFRL DSRC HPC systems. Users are advised to read this guide prior to executing jobs on the system.

1.3 Document Conventions

Unless specified otherwise, % represents the command line prompt. Anything after the % should be entered at the command line prompt in the user's terminal.

1.4 Accessing the System

The SGI Altix 4700 has a total of 9216 cores divided into 18 nodes with 512 cores per node. Only 500 cores per node are available for user jobs. 12 cores per node are reserved for OS-use and are not accessible by the user. The fully qualified hostname of the interactive system is hawk.afrl.hpc.mil.Users are only permitted to login onto the interactive node of the system. The other nodes are for batch jobs only. Users submit their jobs on the front-end node, and the batch system will automatically start their jobs on the other systems based on the load of the system.

1.5 AFRL DSRC Connectivity

Since the SGI Altix 4700 is an integrated component of the AFRL DSRC, user files are Network File System (NFS) mounted from the AFRL DSRC High Availability File Server (HAFS) system to the system. When users log into a system, their home (**$HOME**) directory (which will be the current directory immediately after logging in) physically resides on the file server, but it appears to be local to the system. The AFRL DSRC also supplies archival storage capabilities.

1.6 AFRL DSRC Startup Files

All users are provided a .cshrc and .login file in their home directory. These files reference standard setup files, maintained by the site administrators in a central location, which set up a standard environment for all MSRC users. These files should not be modified.

To set up specific information for your SGI Altix 4700 session, such as environment variables, path information, terminal information or command aliases, place the appropriate commands and information into files called .personal.cshrc and .personal.login. The standard startup files check your home directory for the existence of these files and executes them if found. Commands related to aliases, prompts and some environment variables should go into .personal.cshrc, while commands related to the type of terminal you are using should go into .personal.login. Commands that generate screen output should not be included in your .personal.cshrc file, but should be placed in your .personal.login file.

1.7 The Archive Command

The archive system (**$ARCHIVE_HOME**) is mounted on the interactive node of the system, however, to improve transfer speeds for batch jobs, **$ARCHIVE_HOME** is not mounted to the batch nodes. To transfer files from **$ARCHIVE_HOME**, users will have to use either un-kerberized rcp or the archive command. The archive command is a tool to the AFRL DSRC to help users with transferring files to and from **$ARCHIVE_HOME**. The basic syntax for the archive command is:

archive get [getopts] file1 [file2 ...]

archive put [putopts] file1 [file2 ...]

More information on the archive command can be found online via the archive man page (man archive).

1.8 Additional Information

Much of the information presented in this document is available online through the man pages and is accessible while logged into Hawk by typing:

man {command name}

2.0 AFRL DSRC SGI Altix 4700

This section details the hardware and software available on the SGI Altix 4700 and how they are currently configured.

2.1 Hardware

The SGI Altix 4700 system is a Shared Memory system with 512 cores per node. Each core is a 1.6 GigaHertz (GHz) Itanium2 processor and has a 32 kilobyte (KB) of Level 1 cache (16 mb data/16 mb instruction), 256 KB of Level 2 data cache, 1 megabyte (MB) of Level 2 instruction cache, 9 megabytes (MB) of Level 3 cache and access to either 4 GB or 2 GB (depending on which node the job is run) of system memory. However, due to Linux memory management, only an estimated 875MB of each 1GB is user-accessible.

The SGI Altix 4700 has a total of 9216 cores. There are two cores per 1.6 GHz Itanium2 processor. Each core has a primary data and instruction cache of 16 GB each, secondary data and instruction cache of 32 KB and 1 MB, respectively, and a tertiary, on-chip cache of 9 MB. There is 4 GB of system memory per core for HAWK-1 and HAWK-2. The remaining nodes have 2 GB per core. The interactive node of the SGI Altix 4700 (hawk-0), has the same specifications as above, but is limited to 32 cores.

2.2 File System Overview

Diskspace is subdivided into several areas:

- /small
- /default
- /large

Configuration details of the file system are still being worked as of the publishing date of this guide. To find out specific information regarding the different file systems, please contact the Consolidated Customer Assistance Center (CCAC).

2.2.1 Workspace

Batch jobs are required to run in a local filesystem, generically known as workspace. Each HPC system has high-speed, parallel filesystems for this purpose. Workspace on the SGI Altix 4700 totals 400 TB, and is divided into separate filesystems for administrative and performance reasons. I/O on the filesystems are quicker than I/O on a file system mounted from the network (such as $HOME or $ARCHIVE_HOME). Workspace is intended for the temporary storage of data files needed for your application. This includes (but is not limited to) grid files, restart files, input files and output files. $WRK is to be used rather than the /tmp and /usr/tmp directory areas to prevent possible system crashes. There is no quota on the amount of disk space you may use in workspace. However, a file scrubber is used to automatically remove old files to prevent it from becoming filled. You are encouraged to remove files from workspace when they are no longer needed. The current policy for removing files is on the AFRL DSRC web page and is subject to change based on periodic reviews. The $WRK file system is NOT backed up. In the event of deletion or catastrophic media failure, files and data structures are lost. It is your responsibility to transfer files that need to be saved to a location that allows permanent storage such as **$HOME** or, preferably, **$ARCHIVE_HOME**.

Due to space restrictions on your $HOME directory, it is highly recommended that you use $ARCHIVE_HOME for long-term file storage and backups.

2.2.2 Archival Storage

To provide long-term storage and archiving, the AFRL DSRC provides an archival storage system that combines a large-capacity Serial ATA drive cache, local tape backup/storage and a remote disaster recovery site. This area is referenced using the environment variable **$ARCHIVE_HOME**, is NFS mounted to the interactive node of the SGI Altix 4700, and can be accessed much like any other directory in a filesystem. However, to improve system performance and increase transfer speeds, **$ARCHIVE_HOME** is not NFS mounted to the batch nodes of the SGI Altix 4700. To transfer files in the batch environment, you will need to use unkerberized rcp or the archive command, as shown below:

```
/usr/bin/rcp ${msas}:/msas*/foo/bar
$WRK/username archive get archive_filename local_filename
archive put local_filename
```

For more details about the Archival Storage system, see the Archival Storage User's Guide, located at:

http://www.afrl.hpc.mil/customer/userdocs/

2.3 Operating System

The SGI Altix 4700 runs SUSE Linux Enterprise Server 10, along with SGI ProPack 5SP1 for Linux. This information can be found on the SGI Altix 4700 in:

/etc/SuSE-release and /etc/sgi-release.

2.4 Available Software

Software currently available on the SGI Altix 4700 includes: the Intel and GNU FORTRAN, C and C++ compilers; MPI and many third party software packages.

3.0 Program Development

Program development in the SGI Altix 4700 computing environment is similar to that used in a typical Linux environment. However, the user must take additional steps to utilize the multiple processors available.

3.1 Development Tools

The AFRL DSRC offers many tools to help users who write their own code to develop, compile and debug their software. The SGI Altix 4700 implements the Intel Fortran, C and C++ compilers, along with the GNU Fortran, C and C++ compilers. The Intel FORTRAN compiler is a FORTRAN 90 compiler, whereas the GNU g77 compiler is a FORTRAN 77 compiler. The SGI Scientific Computing Software Library (SCSL) is installed, which provides a collection of mathematical and scientific libraries including Basic Linear Algebra Subprograms (BLAS) levels 1, 2 and 3; LAPACK; Fast Fourier Transforms (FFTs); and convolutions. To aid in debugging your software applications, the AFRL DSRC provides the Intel Debugger and the GNU Debugger. The Intel Debugger and GNU Debugger are symbolic source code debuggers that debug programs compiled by the Intel(R) C/C++ Compiler, the Intel(R) Fortran Compiler, and the GNU compilers (gcc, g++). For full source-level debugging, compile the source code with the compiler option that includes the symbol table information in the compiled executable file. Intel IDB supports DBX and GDB modes. In the GDB mode, Intel Debugger operates like the GNU Debugger, GDB. Recompiling is recommended but not always required when taking a code between systems with different kernels and libraries. The default Intel ifort compiler on both EAGLE and HAWK is version 9.1. The default ifort processor optimization is "-mtune itanium2", which generates code that will run on any Itanium 2 processor. The "-mtune itanium2-p9000" option generates code optimized for the Dual-Core Intel Itanium 2 Processor 9000 Sequence processors, such as is used by the SGI Altix 4700. This only affects instruction order, unless the program uses/executes intrinsics specific to the Dual-Core processor. Documentation for these compilers, libraries and tools is available online in the man page by executing a man on **ifort, icc, icpc, gcc, g++, g77, scsl, gdb or idb**

3.2 Parallel Processing

Users may utilize multiple cores to execute their programs. The compilers are capable of creating parallel programs through the use of compiler directives and parallel standards such as Message Passing Interface (MPI) and OpenMP.

3.2.1 MPI

The goal of MPI is to develop a widely used standard for writing message-passing programs. As such, the interface attempts to establish a practical, portable, efficient and flexible standard for message passing. You can compile MPI programs on the SGI Altix 4700 using the following command line options.

For MPI FORTRAN codes:

ifort -o prog prog.f -lmpi

For MPI C/C++ codes:

icc -o prog prog.c -lmpi

icpc -o prog prog.c -lmpi -lmpi++

Other compiler options are available. Please consult the man pages for the compiler you are using for more information. To execute MPI code on the SGI Altix 4700 you must use the pam or mpirun command.

mpirun -np X

By default, pam starts an MPI rank for each processor core assigned to your PBS job. The **mpirun -np** parameter can be used to specify a different number. This is required if your program uses a non-multiple of 4 cores, or you have requested extra cores to provide additional memory.

More information on MPI can be obtained from:

http://www.mpi-forum.org

3.2.2 OpenMP

OpenMP is a specification for a set of compiler directives, library routines and environment variables that can be used to specify shared memory parallelism in FORTRAN and C/C++ programs. Creating an OpenMP program is done through OpenMP directives in the source code and by adding the -omp flag to your compile string. To run an OpenMP program, you must first tell the program how many threads (processors) to use. This is achieved through the **OMP_NUM_THREADS** environment variable. To set this variable, use the following command in csh (x is the number of cores needed for the job to run):

setenv OMP_NUM_THREADS x

To execute OpenMP code on the SGI Altix 4700 you must use the dplace command.

dplace -x2 -c{core range} {My Program} [Options]

The core range is the range of cores the job needs to run, starting with 0.

For example, if 16 cores are needed to run the job, the core range would be 0-15, and 0-63 for 64 cores, etc. For more information on OpenMP and its directives, please see the following page:

http://www.openmp.org

3.2.3 Shared Memory or SHMEM

Shared Memory, or SHMEM, is a set of libraries that achieve parallelization in a user's code using data passing or one-sided communication techniques. This method of parallelism requires that all processes and threads have access to the same pool of memory. To execute SHMEM code on the SGI Altix 4700 you must use the dplace command.

dplace -x2 -c{core range} {My Program} [Options]

The core range is the range of cores needed to run, starting with 0. For example, if the job needs 16 cores to run, the core range would be 0-15, and 0-63 for 64 cores, etc. For more information on SGI's Altix SHMEM implementation, please see the following page:

http://www.sgi.com/developers/technology/gsm.html

3.2.4 MPI/OpenMP Hybrid Code

Some users are running with codes which are a hybrid of MPI and OpenMP calls. While the SGI Altix 4700 can run such code, there are some issues and limitations with doing so. The OpenMP threads of a hybrid code all need access to the same memory pool. Because of this, hybrid applications are unable to run across more than one node. Unfortunately, hybrid codes will not be able to run on more than 500 cores on the Altix. Since hybrid codes cannot cross nodes, users must use the mpirun command to execute their code since pam may try to execute their code across nodes. The syntax for the SGI Altix 4700's mpirun command is similar to other platforms users are used to running on. Users must also set the OMP_NUM_THREADS variable, much like in the OpenMP section above:

setenv OMP_NUM_THREADS {Number of threads}
mpirun -np {# MPI ranks} omplace -nt {OpenMP threads per MPI rank} {My program}

The -nt option is required if OMP_NUM_THREADS is not set.

3.3 FORTRAN Programming

The default FORTRAN compiler on the SGI Altix 4700 is the Intel ifort compiler. This optimizing and parallelizing compiler can generate a 64-bit code for single, double or extended precision. The default is single.

ifort -o prog prog.f

This command creates an executable called prog. The program is run bytyping the program name at the system prompt.

./prog

Further optimization is available through the use of compiler flags and compiler directives. Please check the ifort and g77 man pages for more details.

3.4 C/C++ Programming

The Intel C and C++ compilers are available on the SGI Altix 4700. These compilers are capable of optimizing and parallelizing code. Compiling a C program on the SGI Altix 4700 is similar to compiling a C program on a typical UNIX system.

icc -o prog prog.c

Compiling a C++ program is also similar.

icpc -o prog prog.cpp

These commands will create an executable program in a file called prog. The program is executed by entering

./prog

Further optimization is available through the use of compiler flags and compiler directives. Please consult the icc or icpc man pages for more details.

3.5 Libraries

3.5.1 Math and Science Libraries

The SGI Altix 4700 has the Intel Math Kernel Librark (Intel MKL), a collection of linear algebra routines, fast Fourier transforms, and vectorized math and random number generation functions. This collection is optimized for use on the SGI Altix 4700. The Intel MKL also provides linear algebra functionality with LAPACK (solvers and ei ensolvers) plus level 1, 2 and 3 BLAS offering the vector, vectormatrix, and matrix-matrix operations needed for complex mathematical software. It is recommended to use the Intel MKL as the math and science libraries on the SGI Altix 4700.

Also available on the SGI Altix 4700 is the SCSL, a collection of LAPACK, collection of solvers for dense linear algebra problems, including linear equations, linear least squares problems, eigenvalue problems, and singular value decomposition problems; Fast Fourier Transforms (FFTs); and convolutions. Both single-threaded and multithreaded routines are available. This library is not automatically included in the link path.

The user must specify the library when linking, as in the following examples:

ifort -o prog prog.f -lmkl

icc -o prog prog.c -lmkl

icpc -o prog prog.c -lmkl

ifort -o prog prog.f -lscs

icc -o prog prog.c -lscs

icpc -o prog prog.c -lscs

4.0 Running Jobs

4.1 Runtime Considerations

The SGI Altix 4700 system allocates more resources to batch jobs than for interactive use. Users will obtain the best throughput for long running or large memory jobs by submitting jobs to the batch queues. Thus, batch use is recommended. Interactive use is allowed, particularly for program development, including debugging and performance improvement, job preparation, job submission and the preprocessing and postprocessing of data. However, only one node on the system is available for interactive use and interactive jobs are limited to 4 cores with 15 minutes of core time per process. Jobs with larger resource requirements must be submitted to the batch queues. When submitting a job, choose the smallest queue that accommodates the job's time and memory requirements. Jobs that request significantly more resources than are actually needed can result in longer wait times and inefficient use of the machine. The workspace filesystems are local to the SGI Altix 4700 via cxfs. Although $HOME is NFS-mounted internally to the compute nodes, I/O access from workspace will be faster than from the HAFS. Thus, it is recommended to keep I/O local to the system. Below is a sample script that copies two input files (one from the $HOME directory and one from the archival storage system) and a program to the user's workspace area, changes to workspace, runs the program, copies two output files (one to the $HOME directory and one to the archival storage system) and then deletes the remaining files:

```
cd $WRK
cp $HOME/small.input$WRK
archive get big.input $WRK
archive get prog $WRK
prog
archive put big.output
mv small.output $HOME
rm small.input big.input prog big.output
```

4.2 Batch Use

PBS Professional, a networked subsystem for submitting, monitoring and controlling a workload of batch jobs on one or more systems, is the batch system for the system. It provides services to monitor queue activity and to delete queued or running jobs. In the event of an orderly system shutdown, PBS jobs will be rerun from the beginning of the job (unless they are specifically marked not to be rerun). More information about PBS is available at: http://www.pbsgridworks.com/Product.aspx?id=1

To allow users to run longer in the queues, a 2 week (336 hour) queue has been implemented and is by request only. To keep the queue structure fair to all users, several restrictions have been put into place:

- The primary resource to schedule jobs by is the core hour (CPH). This quantity is equal to the wall time requested multiplied by the number of cores, as shown below, and cannot exceed a value of 750,000 per user. Table 1 below shows the formula used, and some examples.

  (ncores * walltime) = CPH

# of Jobs	# of Cores	Walltime	Total CPH
1	1000	48	48,000
2	256	48	24,576
25	8	96	19,200

- A user can use, at most, 3000 cores for MPI jobs and 500 cores for SMP and MPI/OpenMP hybrid jobs.

- The maximum wall time of any queue shall not exceed 336 hours.

- A background job cannot start if there is a foreground job in any queue.

The list of queues and the upper limits of job resources for these queues are available on the web at:

http://www.afrl.hpc.mil/overall/policy_procedure/policies/use_policy.php

These limits are subject to change based on periodic review of system utilization and system configuration.

TIP: Because of CPH, it is not recommended that users always submit jobs with the maximum queue runtime. If the job requires less time to run than the queue default, request the smaller of the two. This will result in a lower CPH and will allow the job to run more quickly.

Example:
User Joe submits a 4 core job using the queue default walltime of 168 hours. This results in a CPH of 672. User Bob submits the same 4 core job, but only he requests the 120 hours he needs to finish the job. His CPH would be 480.

This allows user Bob to submit twice (2.8 to be exact) as many jobs as Joe in the same amount of CPH. If user Bob can alter his walltime and fit his jobs within 110 hours, he can submit 3 times the number of jobs as Joe. This also has a domino effect on the queue structure, resulting in faster throughput throughout the entire system.

4.2.1 Queueing Structure

The AFRL DSRC SGI Altix 4700 has three queues: debug, standard and background. These queues are available 24 hours a day 7 days a week. The debug queue accepts jobs that require up to 32 cores, 1 hour of core time and 56 GB of memory. This queue is intended for short runs. The standard queue is divided into subqueues, but users do not submit jobs directly to these subqueues. Rather, the user specifies the number of cores, the core time and memory requirements using the bsub options below. The job is then routed to the appropriate subqueue. The background queue is also available for production work. Jobs run in the background queue are not charged against a user's allocation. However, jobs in the background queue are only started when utilization of the machine is low and never when foreground jobs are waiting.

4.2.2 Preparing Jobs

Before a user submits a job, they should prepare a job script. A job script is a UNIX shell script that contains all the commands the user will execute during the job. PBS will place the error and output files in the directory the job was submitted from, so scripts must be written with this in mind. To avoid wasting resources, if the job uses a large number of cores, it is best to archive large output files interactively or in a separate batch job.

Here is a sample serial job script:

```
#!/bin/csh
#PBS -l walltime=h:mm:ss
#PBS -l select=ncpus=1
#PBS –l resource=x
#PBS -l job_type={SMP|MPI|MIX}
#PBS -j oe
#PBS -m abe
#PBS -M user1@afrl.hpc.mil, user2@your.domain
#PBS -A account_number (16-character account number)
#
#  Change to $WORK_DIR directory and copy input file.
#  We're using the archive command to pull information from
#  $ARCHIVE_HOME. The -C option tells the archive command which
#  directory in $ARCHIVE_HOME to look into. If you had a file in
```

```
#  $ARCHIVE_HOME/foo/bar, you would give the -C option foo/bar.
#
cd $WORK_DIR
archive get -C {directory in $ARCHIVE_HOME} {filename}
#
#  Run the analysis.
date
./my_job
date
#
#  Archive output. First we tar up $WORK_DIR to compress
#  the data and make it easier to transfer, then we use the
#  archive command to push it to $ARCHIVE_HOME. The -C option works
#  the same way here as it did above, except in this case,
#  you would give it a path to where you wanted to store
#  the files. More information on the archive command can
#  be found in the archive man page (man archive).
#
tar cvf ../{output filename}.tar .
archive put -C {directory in $ARCHIVE_HOME} ../{output filename}.tar
#
#  Exit the script.
```

This script copies an input file and a program to the user's $WORK_DIR directory. The $WORK_DIR directory is a directory created by PBS for users to run their batch jobs. This directory has certain protections from the workspace scrubber, as long as the job is running. Plus, five days after it finishes, this directory will not be removed. Then the script changes to the $WORK_DIR directory, runs the program, copies two output files to permanent storage (one to the $HOME directory and one to the archival storage system $ARCHIVE_HOME), and then deletes the remaining files, the $WORK_DIR directory and exits.

NOTE: $WORK_DIR only exists in PBS, you will not be able to change to that directory using the variable $WORK_DIR.

4.2.3 Submitting Jobs

Once a job script is prepared, the qsub command is used to submit the script to PBS. The command has the following syntax:

```
qsub  script
```

Some important PBS options used are (type man qsub for a complete list of options available):

PBSPro Command	Option Meaning
-Q	Show Queue information.
-Qf	Show Queue information.
tracejob	Job history
qstat -xf	Job history
qstat -a	Show all jobs currently running.
qhold	Hold a job.
qrls	Resume a job placed on hold by a user.
qdel	Kill a job.

When a job is submitted to PBS, a unique identifier is assigned to the job by the batch system similar to below:

2079.hawk-0.afrl.hpc.mil

This unique identifier is needed when deleting a job. Options of qsub commands are specified within the script file itself.

The options are specified using syntax similar to LSF, but each line that contains an option must begin with the #PBS string. Options that are specified within the script file must precede the first executable shell command of the file as in the following example.

```
#!/bin/csh
#PBS -l walltime=h:mm:ss
#PBS -l select=ncpus=1
#PBS -l resource=x
#PBS -l job_type={SMP|MPI|MIX}
#PBS -j oe
#PBS -m abe
#PBS -M user1@afrl.hpc.mil, user2@your.domain
#PBS -A account_number (16-character account number)**
```

**This is an example account number. To find your account number, check your $ACCOUNT variable using

echo $ACCOUNT

More sample batch scripts can be found at the following URL:

http://www.afrl.hpc.mil/customer/userdocs/samples/samplebatch.php

4.2.4 Monitoring Jobs

The qstat command is used to report the status of the batch jobs that are currently queued or running. Type man qstat for information about qstat and the options that are available. The qstat command lists all jobs that are running and queued.

% qstat

JOBID	Name	USER	Time	S	QUEUE
18426	XXXXX	user1	0	H	standard
27441	XXXXX	user2	1803:00:36	R	standard

Table 2 is an output guide for the qstat command.

Header	Meaning
JOBID	A unique identifier that consists of the original request number.
Name	Name of the job.
User	Username of the person submitting the job.
Time Use	CPU Time used by the job.
S	Job State (Hold, Running, Queued, etc)
Queue	The queue in which the job resides.

4.2.5 Deleting Jobs

In PBS, queued or running jobs are removed using the qdel command. The syntax is:

% qdel jobid

Example:

The below example will show the listing from qstst (to examine currently queued jobs), and bkill to kill the job.

```
% qstat -u ‘echo $USER’

Job id          Name          User            Time Use S Queue

----------------  ---------------- ----------------  -------- - -----

3373    XXXX    XXXX              0 Q standard

% qdel 3373
```

5.0 Customer Service

5.1 Consolidated Customer Assistance Center (CCAC)

For customer assistance, call the Consolidated Customer Assistance Center (CCAC) at:

Toll Free: 1-877-CCAC-039 (1-877-222-2039)
Local: (937) 255-0679
DSN: 785-0679
e-mail: help@ccac.hpc.mil.

If you have any questions regarding the AFRL DSRC, contact CCAC first. If your problem or question is beyond the scope of the CCAC, they will refer you to the appropriate resource.

5.2 AFRL DSRC Support

In-depth technical inquiries and problems are forwarded to the AFRL DSRC Customer Assistance and Technology Center (CATC), which pursues such inquiries and problems through resolution as rapidly as possible. The AFRL DSRC CATC will attempt to determine the nature of the problem, then identify and coordinate whatever resources are needed to resolve the problem.

5.3 AFRL DSRC Website

The AFRL DSRC website (http://www.afrl.hpc.mil) is the best source for current AFRL DSRC information. Some of the topics found on the website include:

- **APPLICATIONS (**http://www.afrl.hpc.mil/software/)

  Short and long descriptions of current AFRL DSRC applications

- **SYSTEMS (**http://www.afrl.hpc.mil/hardware/)

  Information on AFRL DSRC servers and Archival Storage

- **CONSOLIDATED CUSTOMER ASSISTANCE CENTER (CCAC) (http://www.ccac.hpc.mil)**

  Available CCAC Services

- **ONLINE DOCUMENTATION (**http://www.afrl.hpc.mil/customer/userdocs/)

  Listings of the AFRL DSRC User Guides are available for viewing.

  Instructions are given on obtaining postscript versions.

- **TRAINING (https://okc.erdc.hpc.mil/index.jsp)**

  Current course offerings and schedule

- **FREQUENTLY ASKED QUESTIONS**

  Read about various topics (such as “Customizing Your Environment”)

- **POLICIES AND PROCEDURES (**http://www.afrl.hpc.mil/overall/policy_procedure/)

  The latest policies regarding usage of the AFRL DSRC resources